Dieter Schuller and Jan Sürmeli: *Dienstgüte-basierte Service-Selektion für Zustandsbehaftete Services.* In: Proceedings of the 3rd Central-European Workshop on Services and their Composition (ZEUS 2011), Karlsruhe, Germany, vol. 705, p. 42-49, February 2011. ISBN ISSN 1613-0073. http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-705/

Dienstgüte-basierte Service-Selektion für Zustandsbehaftete Services

Dieter Schuller[1] and Jan Sürmeli[2]

[1] Multimedia Communications Lab (KOM),
Technische Universität Darmstadt, Germany
Dieter.Schuller@KOM.tu-darmstadt.de
[2] Institut für Informatik,
Humboldt-Universität zu Berlin, Germany
suermeli@informatik.hu-berlin.de

Zusammenfassung In Serviceorientierten Architekturen können Geschäftsprozesse durch Komposition von lose gekoppelten Services realisiert werden. Sind entsprechende Services auf Service-Marktplätzen vorhanden, kann ein Service-Konsument zwischen Services wählen, die die von ihm benötigte Funktionalität bereitstellen, basierend auf deren Dienstgüte (engl.: Quality of Service – QoS). Diese QoS-basierte Service-Selektion wird in der Literatur zumeist für Zustandslose Services durchgeführt. Zustandsbehaftete Services können, je nach tatsächlichem Ausführungspfad, verschiedene Ausprägungen in ihren QoS-Attributen annehmen, was in verwandten Arbeiten bisher nicht berücksichtigt wird. In der vorliegenden Arbeit werden Ansätze skizziert, mit denen die QoS-basierte Service-Selektion auch für Zustandsbehaftete Services durchgeführt werden kann.

1 Einleitung

In hoch kompetitiven Märkten, in denen die agierenden Unternehmen ähnliche Produkte und Services anbieten (wie bspw. in der Finanzindustrie), ist es erforderlich, dass die Geschäftsprozesse effizient ausgeführt werden. Zudem müssen Unternehmen in solchen Märkten in der Lage sein, ihre Geschäftsprozesse dynamisch und flexibel an sich ändernde, marktgetriebene Rahmenbedingungen anzupassen. Diese Flexibilität können Unternehmen durch die Umsetzung einer Serviceorientierten Architektur (SOA) erreichen, in der lose gekoppelte Services – mit einer mehr oder weniger grob granularen Funktionalität (vgl. [1]) – für die Realisation der Unternehmenseigenen Geschäftsprozesse eingesetzt werden. In diesem Zusammenhang – um agile Geschäftsprozesse zu ermöglichen und sie zu unterstützen – wird das SOA Paradigma häufig empfohlen [2].

Um die Realisation der Geschäftsprozesse möglichst effizient zu gestalten, sollten diejenigen Services eingesetzt werden, die sowohl die notwendigen Funktionalitäten bereitstellen als auch den qualitativen Anforderungen der Unternehmen genügen. Diese Services müssen dabei nicht notwendigerweise ausschließlich im eigenen Unternehmen vorhanden sein. Sind (aus funktionaler Sicht) entsprechende Services auf Service Marktplätzen verfügbar – wie in der Vision des

The documents distributed by this server have been provided by the contributing authors as a means to ensure timely dissemination of scholarly and technical work on a non-commercial basis. Copyright and all rights therein are maintained by the authors or by other copyright holders, not withstanding that they have offered their works here electronically. It is understood that all persons copying this information will adhere to the terms and constraints invoked by each author's copyright. These works may not be reposted without the explicit permission of the copyright holder.

Internet of Services postuliert – können Unternehmen zwischen Services (die die benötigte Funktionalität bereitstellen) basierend auf deren Qualität (engl.: Quality of Service – QoS) wählen. Dieses Service-Selektions-Problem (SSP) wurde in der Literatur bereits in vielen Arbeiten (bspw. in [3–5]) adressiert, jedoch lediglich für Zustandslose Services gelöst. Bei Zustandsbehafteten Services sind die zur Laufzeit aufgerufenen Operationen innerhalb der Services im Vorhinein unbekannt, sodass zur Planungszeit (Zeitraum, in dem das SSP gelöst wird) nicht mit Sicherheit gesagt werden kann, welche QoS-Attribute die einzelnen Services zur Laufzeit haben werden. Dieses Problem wird in der vorliegenden Arbeit adressiert.

Im folgenden Abschnitt 2 wird die vorliegende Arbeit von verwandten Arbeiten abgegrenzt. In Abschnitt 3 wird das SSP (für Zustandslose Services) vorgestellt, für dessen Lösung feste (eindeutige) QoS-Werte erforderlich sind. Die besonderen Eigenschaften für Zustandsbehaftete Services werden anschließend in Abschnitt 4 beschrieben. Darauf aufbauend skizziert Abschnitt 5 mögliche Ansätze für das SSP mit Zustandsbehafteten Services. Abschnitt 6 dient der Zusammenfassung des vorliegenden Beitrags sowie der Vorstellung von Ideen für zukünftige Arbeiten.

2 Verwandte Arbeiten

Das SSP (für Zustandslose Services) wurde in der Literatur bereits von vielen Autoren adressiert. In einigen Arbeiten (bspw. in [3, 6, 7]) werden heuristische Lösungsansätze vorgeschlagen. Ansätze, die auf eine optimale Lösung des SSP abzielen, werden in [8,9] beschrieben. Um das SSP für komplexe Workflowpatterns zu lösen, wird bei diesen Arbeiten für jeden möglichen (sequenziellen) Ausführungspfad eine Lösung mithilfe von Standardmethoden (wie Branch & Bound) aus dem Bereich des Operations Research [10] erstellt. Dies umfasst auch Zyklische Strukturen, sodass solche Strukturen zunächst offen gelegt werden müssen. Insofern ist Kenntnis und Berücksichtigung aller möglichen Ausführungspfade, deren Anzahl mit jeder zusätzlichen Verzweigung exponentiell ansteigt, für die Berechnung einer optimalen Lösung Voraussetzung.

Der vorliegende Ansatz zielt ebenfalls auf die Berechnung einer (nahezu) optimalen Lösung für das SSP ab. Die Kenntnis aller möglichen Ausführungspfade wird dabei jedoch nicht benötigt, wodurch sich der vorliegende Ansatz von verwandten Arbeiten in diesem Bereich abgrenzt. Des Weiteren können auch rekursive Verschachtelungen von Workflowpatterns berücksichtigt werden, was nach unserem Wissensstand in verwandten Arbeiten nicht adressiert wurde.

3 Service-Selektions-Problem

Bei der QoS-basierten Service-Selektion geht es darum, für einen abstrakten Workflow (bspw. in Business Process Modeling Notation – BPMN) konkrete Services zu finden, die die einzelnen Workflow-Schritte (d. h., die abstrakten Services) realisieren. Das Ergebnis der Service-Selektion stellt dabei einen Ausführungsplan

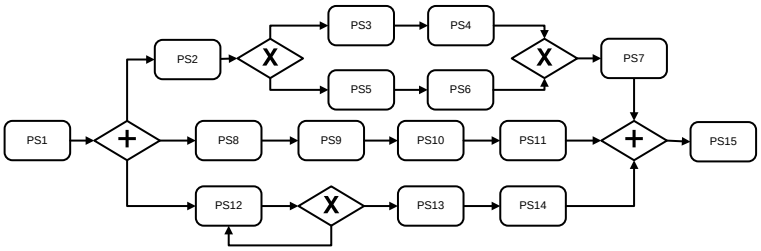

Abbildung 1: Beispiel für einen Workflow

dar, in dem eine Zuordnung von konkreten Services zu den einzelnen abstrakten Services vorgenommen wird.

Um ein solches SSP spezifizieren und lösen zu können, müssen die QoS-Attribute der infrage kommenden Services entsprechend ihrer Anordnung im Workflow aggregiert werden. In der vorliegenden Arbeit berücksichtigen wir dabei die Workflowpatterns *Sequenz*, *AND-Split/-Join*, *XOR-Split/-Join*, die in [12] beschrieben sind, sowie *Simple Loop* (vgl. [13]). Ein Beispiel für einen solchen Workflow ist in Abbildung 1 gegeben. Die Workflowschritte sind in dieser Abbildung mit PS abgekürzt.

Die Menge der abstrakten Services bezeichnen wir mit I, $i \in I = \{1, ..., n\}$. Jedem abstrakten Service wird exakt ein konkreter Service $j \in J_i = \{1, ..., m_i\}$ zugeordnet. Dabei geben die Entscheidungsvariablen $x_{ij} \in \{0, 1\}$ wider, ob ein konkreter Service j einem abstrakten Service i zugeordnet ist. Als nicht-funktionale bzw. QoS-Parameter verwenden wir Ausführungszeit e (benötigte Zeit, um einen Service auszuführen), Kosten c (Kosten für die Invokation eines Services), Zuverlässigkeit r (Wahrscheinlichkeit, dass der Service erfolgreich ausgeführt wird), sowie Durchsatz d (Anzahl paralleler Service Invokationen). Mit diesen Parametern lassen sich die Aggregationstypen Summation, Multiplikation sowie Min/Max-Operator abdecken, sodass weitere QoS-Parameter, die zu diesen Aggregationstypen gehören, leicht eingefügt werden können. Bezüglich möglicher Verzweigungen definieren wir die Menge L der Pfade l als $l \in L = \{1, ..., l^{\#}\}$. D. h., l stellt die entsprechende Pfad-Nummer innerhalb einer Verzweigung dar. Die AND-Verzweigung nach $PS1$ erzeugt drei Pfade l, d. h. $L = \{1, 2, 3\}$. Die Menge der abstrakten Services innerhalb einer Verzweigung L wird mit $IW_L \subseteq I$ bezeichnet. $IW_l \subseteq IW_L$ stellt die Menge der abstrakten Services eines Pfads l dar. Das Ergebnis der Service-Selektion, d. h. die Menge der selektierten Services, wird durch S dargestellt. Gibt es bei dem betrachteten Workflow mehrere Ausführungsmöglichkeiten bzw. mehrere Ausführungspfade (wie bspw. bei dem XOR-Split nach $PS2$ in Abbildung 1), ist zur Planungszeit nicht bekannt, welcher der möglichen Pfade ausgeführt wird.

In einer Average-Case Analyse werden Wahrscheinlichkeiten p_l für mögliche Pfade l angenommen, die angeben, mit welcher Wahrscheinlichkeit (bei einem XOR-Split) ein bestimmter Pfad ausgeführt wird, und bei Berechnung einer

QoS	**Sequenz**	**AND-Split/-Join**	**XOR-Split/-Join**	**Loop**
e	$\sum_{i \in IS} \sum_{j \in J_i} e_{ij} x_{ij}$	$\max_{l \in L}(\sum_{i \in IW_l} \sum_{j \in J_i} e_{ij} x_{ij})$	$\sum_{l \in L} p_l \sum_{i \in IW_l} \sum_{j \in J_i} e_{ij} x_{ij}$	$\frac{1}{1-\rho_i} e_{ij}$
c	$\sum_{i \in IS} \sum_{j \in J_i} c_{ij} x_{ij}$	$\sum_{l \in L} \sum_{i \in IW_l} \sum_{j \in J_i} c_{ij} x_{ij}$	$\sum_{l \in L} p_l \sum_{i \in IW_l} \sum_{j \in J_i} c_{ij} x_{ij}$	$\frac{1}{1-\rho_i} c_{ij}$
r	$\prod_{i \in IS} \sum_{j \in J_i} r_{ij} x_{ij}$	$\prod_{l \in L} \prod_{i \in IW_l} \sum_{j \in J_i} r_{ij} x_{ij}$	$\sum_{l \in L} p_l \prod_{i \in IW_l} \sum_{j \in J_i} r_{ij} x_{ij}$	$\frac{(1-\rho_i) r_{ij}}{1-\rho_i r_{ij}}$
d	$\min_{i \in IS}(\sum_{j \in J_i} d_{ij} x_{ij})$	$\min_{l \in L}(\min_{i \in IW_l}(\sum_{j \in J_i} d_{ij} x_{ij}))$	$\sum_{l \in L} p_l \min_{i \in IW_l}(\sum_{j \in J_i} d_{ij} x_{ij})$	d_{ij}

Tabelle 1: Average-Case Aggregationsfunktionen

Lösung für das SSP berücksichtigt. Hierfür verwenden wir die Aggregationsfunktionen in Tabelle 1, die in unserer Arbeit in [14, 15] erläutert werden. Die Berücksichtigung von Loops ist ebenfalls dort beschrieben. Die berechnete Lösung (als Ergebnis der Optimierung) spiegelt dann den Mittelwert über alle Pfade wider. Insofern werden gegebene untere bzw. obere Schranken (Restriktionen für die QoS-Parameter) lediglich (rechnerisch) im Durchschnitt eingehalten. Die Durchführung einer solchen Average-Case-Analyse wird in unserer Arbeit in [14] vorgestellt und daher in der vorliegenden Arbeit nicht weiter beschrieben.

Bei einer Worst-Case-Analyse betrachtet man im Unterschied zu der Average-Case-Analyse für jeden QoS-Parameter den schlechtest möglichen Pfad – bspw. den Pfad mit der größten aggregierten Ausführungszeit oder mit der geringsten Zuverlässigkeit. Dies können durchaus unterschiedliche Pfade für die verschiedenen QoS-Parameter sein. Das Ergebnis einer solchen Service-Selektion für den Worst-Case wäre ein Ausführungsplan, der die QoS-Restiktionen für den gesamten Workflow auf keinen Fall verletzt. Bei einer Best-Case-Analyse wird analog der Worst-Case-Analyse für jeden QoS-Parameter der best mögliche Pfad bei der Optimierung berücksichtigt. Das bedeutet, dass für die jeweils anderen Pfade die Restriktionen nicht berücksichtigt und insofern (höchstwahrscheinlich) verletzt werden. Das Ergebnis einer solchen Service-Selektion führt zu einem Ausführungsplan, bei dem die Zielfunktion den best möglichen Wert im Vergleich zur Worst-Case und Avg-Case-Analyse aufweist, jedoch werden die Restriktionen möglicherweise in keinem der möglichen Ausführungspfade eingehalten. Bezüglich der in Tabelle 1 angegebenen Average-Case-Aggregationsfunktionen würden sich bei einer Worst-/Best-Case-Analyse lediglich die Funktionen für XOR-Split/-Join ändern – wie in [15] beschrieben – und es würden untere (Best-Case) bzw. obere (Worst-Case)-Schranken für die Anzahl an Iterationen bei Loops angenommen. Diese (geänderten) Funktionen werden dann für die Optimierung des SSPs herangezogen.

Unabhängig davon, welche dieser drei möglichen Analysen schlussendlich durchgeführt wird, würde bei der Optimierung des SSPs (wie in [14]) davon ausgegangen, dass für jeden QoS-Parameter eines Services genau ein Wert existiert. Werden jedoch Zustandsbehaftete Services betrachtet, kann diese Annahme nicht getroffen werden. Stattdessen stehen hier Wertebereiche für die einzelnen QoS Werte zur Verfügung, was im folgenden Abschnitt erläutert wird.

4 Zustandsbehaftete Services

Ein Zustandsbehafteter Service ist ein offenes System mit eigenem Kontrollfluss und eigenen Zuständen. Ein Zustand umfasst wie bei einem abgeschlossenen System den aktuellen Wert aller Systemgrößen. Der Kontrollfluss ändert den Zustand des Systems basierend auf dem aktuellen Zustand: Es werden Entscheidungen basierend auf Systemgrößen oder nichtdeterministisch getroffen. Zudem werden besagte Systemgrößen manipuliert. Der Unterschied zu einem abgeschlossenen System besteht darin, dass ein Service mit seiner Umwelt kommuniziert. Diese Umwelt kann wiederum aus mehreren Services bestehen. So kann beispielsweise ein Service zur Buchung einer Reise an einen Service für die Hotelsuche und einen Service zur Flugsuche gebunden sein. Die Ein- und Ausgabedaten eines Zustandsbehafteten Services sind also Ein- und Ausgabeströme von Nachrichten der Umwelt. Die Reaktion des Services auf eine bestimmte Nachricht seiner Umwelt ist stets abhängig von seinem inneren Zustand: So wird beispielsweise eine bestimmte Anfrage a erst genehmigt, wenn von der Umwelt vorher bereits Nachricht b gesendet wurde. An seinem inneren Zustand kann der Service erkennen, ob b bereits empfangen wurde oder nicht.

Da zur Planungszeit das genaue Verhalten der Umwelt nicht bekannt ist, also nicht vorausgesagt werden kann, welche Nachrichten genau an den Service gesendet werden, können wir auch nicht genau sagen, welcher Pfad des Kontrollflusses tatsächlich während der Ausführung gewählt wird. Daher ist auch zur Planungszeit nicht klar, welche Operationen wie häufig und in welcher Reihenfolge ausgeführt werden. Dies macht die Angabe von fixen QoS-Werten für einen Zustandsbehafteten Service in der Regel unmöglich.

Es ist jedoch möglich, Wertebereiche V für jeden einzelnen QoS-Parameter anzugeben (bspw. V_e, V_c, V_r, V_d), die jeden möglichen Pfad des Kontrollflusses abdecken. Die einfachste Variante dafür ist ein Intervall, das eine untere und eine obere Grenze für jeden QoS-Parameter angibt. In einigen Fällen ist es schwierig oder gar unmöglich, eine genaue obere oder untere Grenze für einen Parameter anzugeben. In diesem Falle muss entweder approximiert oder das fehlende Wissen über eine Schranke als offenes Intervall kodiert werden. Alternativ zu Intervallen können wir uns bspw. Paare aus Erwartungswert und Standardabweichung vorstellen. Das Finden solcher genauen oder approximierten Wertebereiche kann auf unterschiedliche Art und Weise geschehen. Eine Variante ist es, das Modell des Services zu analysieren [16]. Dabei wird jedes mögliche Verhalten der Umwelt in Betracht gezogen; das Ergebnis ist ein Intervall. Ohne weiteres Wissen über das bevorzugte Verhalten der Umgebung ist es nicht möglich, die Werte im Intervall zu gewichten. Eine weitere Variante wäre Monitoring, wobei alle Ausführungen des Services aufgezeichnet werden (vgl. [17]). Anhand dieser gespeicherten Informationen können möglicherweise Rückschlüsse auf nachfolgende Ausführungen getroffen werden. Die Kombination beider Verfahren würde die Gewichtung der Werte innerhalb des Intervalls erlauben.

5 Service-Selektion bei Zustandsbehafteten Services

Stehen keine eindeutigen QoS-Werte für die einzelnen Services zur Verfügung sondern Wertebereiche, gestaltet sich der Vergleich zweier Services bezüglich ihrer QoS schwieriger. Dies wird an folgendem Beispiel verdeutlicht: Kostet ein Service S1 bspw. fest 2ct pro Invokation und benötigt er dabei fest 3s (Ausführungszeit), so können diese Werte mit denjenigen von Service S2 verglichen werden, der bspw. fest 3ct kostet bei einer festen Ausführungszeit von 2s. Hinsichtlich Kosten wäre S1 besser als S2, hinsichtlich Ausführungszeit wäre S2 zu bevorzugen. Über die individuelle Präferenz des Nutzers hinsichtlich der verschiedenen QoS-Parameter kann dann entschieden werden, welcher Service ausgewählt werden soll. Sind jedoch Wertebereiche anstatt fester Werte für jeden QoS-Parameter gegeben, ist diese Entscheidung nicht mehr trivial. Zwei Services können ohne Weiteres nicht miteinander verglichen werden. Insofern kann auch kein Ranking für die Kandidaten von Services für eine abstrakte Aktivität erstellt werden. Für die Service-Selektion müssen Services jedoch miteinander verglichen werden können.

Im Folgenden werden daher Ansätze skizziert, um das SSP bei Vorliegen von Wertebereichen für einzelne QoS-Parameter zu lösen. Für jeden Ansatz wird eine Annahme bezüglich der Spezifikation des Wertebereichs getroffen. Für den jeweils erstellten Ausführungsplan können die QoS für den gesamten Workflow zwar nicht eindeutig bestimmt werden, sie haben allerdings je nach Verfahren bestimmte Eigenschaften.

5.1 Worst QoS – Worst Case

Für dieses Verfahren wird angenommen, dass der Wertebereich in Form eines geschlossenen Intervalls vorliegt. Für jeden Service wird für jeden QoS-Parameter der jeweils schlechteste Wert – Worst QoS – für das Optimierungsproblem herangezogen. Der schlechteste Wert ist für jeden QoS-Parameter entweder die linke oder rechte Grenze des Intervalls: $e := max(V_e)$, $c := max(V_c)$, $r := min(V_r)$, $d := min(V_d)$. Um hier tatsächlich eine Worst-Case Abschätzung zu erhalten, schlagen wir vor, eine Worst-Case-Analyse durchzuführen (vgl. Abschnitt 3). Ergebnis dieser Worst-Case-Analyse ist ein Ausführungsplan. Um nun dem Umstand Rechnung zu tragen, dass keine fixen QoS Werte sondern lediglich QoS Intervalle haben, werden in einem zweiten Schritt für die erhaltene Selektion von Services QoS-Intervalle VW_e, VW_c, VW_r, VW_d für den gesamten Workflow berechnet. Hierfür werden die unteren und oberen Schranken durch Aggregation der jeweils schlechtesten bzw. besten Werte der jeweiligen QoS-Intervalle (der selektierten Services) bestimmt. D. h., $min(VW_e) = Aggregate(min(V_{e_j})|\forall j \in S)$ und $max(VW_e) = Aggregate(max(V_{e_j})|\forall j \in S)$, wobei sich $Aggregate$ auf die Verwendung der entsprechenden Aggregationsfunktion in Tabelle 1 bezieht. Die Berechnung der entsprechenden Intervalle für die anderen QoS-Parameter erfolgt analog. Somit wird ein Ausführungsplan erstellt, für den die gegebenen QoS-Restiktionen stets eingehalten werden und für den die aggregierten QoS-Werte in den entsprechenden Intervallen $[min(VW), max(VW)]$ liegen.

Zusammenfassend sei an dieser Stelle noch mal erwähnt, dass für die Durchführung der Worst-Case-Analyse die untere/obere QoS Intervallgrenze der einzelnen Services herangezogen wird, um diese Services für die Optimierung miteinander vergleichen zu können, was (wie oben erwähnt) eine Voraussetzung zur Durchführung der Optimierung darstellt. Für die Berechnung der zu erwartenden QoS für den gesamten Workflow werden anschließend sowohl die oberen als auch unteren QoS Intervallgrenzen der ausgewählten Services berücksichtigt, um auf diese Weise das ganze QoS Intervall eines Services abdecken zu können.

5.2 Average QoS – Average Case

Ein zweiter möglicher Lösungsansatz funktioniert unter der Annahme, dass für jeden QoS-Parameter der Wertebereich in Form von Mittelwert und Standardabweichung vorliegt. Diese können bspw. durch Simulation oder Monitoring gefunden werden (vgl. Abschnitt 4). Es bietet sich die Durchführung einer Average-Case-Analyse an (vgl. Abschnitt 3), für die die besagten Mittelwerte der einzelnen QoS-Parameter herangezogen würden. Das Ergebnis wäre ein Ausführungsplan, der die Restriktionen im Durchschnitt erfüllt. Mithilfe der Standardabweichungen für die QoS-Werte der selektierten Services oder durch Simulation ließe sich ebenfalls ein Intervall für die QoS-Werte des gesamten Workflows gemäß des errechneten Ausführungsplans generieren.

5.3 Best QoS – Best Case

Analog zu Abschnitt 5.1 könnten für die Service-Selektion auch die jeweiligen besten Werte aus den QoS-Intervallen ($e := min(V_e)$, $c := min(V_c)$, $r := max(V_r)$, $d := max(V_d)$) herangezogen werden. Mit diesen würde eine Best-Case-Analyse durchgeführt. Basierend auf dem berechneten Ausführungsplans ließen sich analog zu Abschnitt 5.1 Intervalle für die QoS-Werte auf Workflow-Ebene bestimmen. Jedoch ist bei Durchführung einer Best-Case-Analyse sowie durch die Wahl der jeweils besten Werte für die QoS-Parameter die Wahrscheinlichkeit eher hoch, dass bei der Ausführung des auf diese Weise erstellten Ausführungsplans die QoS-Restriktionen verletzt werden.

6 Zusammenfassung und Ausblick

Das SSP wird in der Literatur in vielen wissenschaftlichen Arbeiten diskutiert. Nach unserem Wissensstand wurden hier jedoch stets Zustandslose Services berücksichtigt. In der vorliegenden Arbeit wurde das SSP mit Zustandsbehafteten Services adressiert. Die Herausforderung besteht dabei darin, dass für die infrage kommenden Services keine eindeutigen QoS-Werte existieren (vgl. Abschnitt 4). In unserer Arbeit in [16] haben wir diesbezüglich ein Verfahren entwickelt, mit dem sich Intervalle für die QoS-Werte approximieren lassen. Unter Verwendung dieser Intervalle wurden in Abschnitt 5 mögliche Ansätze für die Lösung des SSP mit Zustandsbehafteten Services skizziert. Unsere zukünftige Arbeit zielt auf die Umsetzung und Evaluation dieser Ansätze sowie auf deren Kombination ab.

Acknowledgements. Diese Arbeit wurde in Teilen durch das E-Finance Lab e. V., Frankfurt am Main, Deutschland, (http://www.efinancelab.de) unterstützt.

Literatur

1. Krafzig, D., Banke, K., Slama, D.: Enterprise SOA: Service-Oriented Architecture Best Practices. Prentice Hall PTR, Upper Saddle River, NJ, USA (2004)
2. Papazoglou, M.P.: Service-Oriented Computing: Concepts, Characteristics and Directions. In: Web Information Systems Engineering. (2003) 3–12
3. Anselmi, J., Ardagna, D., Cremonesi, P.: A QoS-based Selection Approach of Autonomic Grid Services. In: International Conference on Service-oriented Computing. (2007) 1–8
4. Menascé, D.A., Casalicchio, E., Dubey, V.: A Heuristic Approach to optimal Service Selection in Service-oriented Architectures. In: Workshop on Software and Performance. (2008) 13–24
5. Huang, A.F.M., Lan, C.W., Yang, S.J.H.: An optimal QoS-based Web Service Selection Scheme. Information Sciences **179**(19) (2009) 3309–3322
6. Jaeger, M.C., Rojec-Goldmann, G.: SENECA-Simulation of Algorithms for Selection of Web Services for Composition. In: Technologies for E-Services. (2005) 84–97
7. Mabrouk, N.B., Georgantas, N., Issarny, V.: A semantic end-to-end QoS Model for dynamic Service oriented Environments. In: Proceedings of PESOS. (2009) 34–41
8. Ardagna, D., Pernici, B.: Adaptive Service Composition in Flexible Processes. IEEE Trans. Software Eng. **33**(6) (2007) 369–384
9. Zeng, L., Benatallah, B., Ngu, A.H., Dumas, M., Kalagnanam, J., Chang, H.: QoS-Aware Middleware for Web Services Composition. Transactions on Software Engineering **30**(5) (2004) 311–327
10. Domschke, W., Drexl, A.: Einführung in Operations Research. Springer Verlag, Heidelberg (2007)
11. Van Der Aalst, W.M.P., Ter Hofstede, A.H.M., Kiepuszewski, B., Barros, A.P.: Workflow Patterns. Distributed Parallel Databases **14**(1) (2003) 5–51
12. Cardoso, J., Sheth, A.P., Miller, J.A., Arnold, J., Kochut, K.: QoS for Workflows and Web Service Processes. Journal of Web Semantics **1**(3) (2004) 281–308
13. Schuller, D., Miede, A., Eckert, J., Lampe, U., Papageorgiou, A., Steinmetz, R.: Qos-based optimization of service compositions for complex workflows. In: International Conference on Service Oriented Computing. (2010) 641–648
14. Schuller, D., Eckert, J., Miede, A., Schulte, S., Steinmetz, R.: QoS-Aware Service Composition for Complex Workflows. In: International Conference on Internet and Web Applications and Services. (2010) 333–338
15. Gierds, C., Sürmeli, J.: Estimating costs of a service. In: Central-European Workshop on Services and their Composition. (2010) 121–128
16. Repp, N., Schuller, D., Siebenhaar, M., Miede, A., Niemann, M., Steinmetz, R.: On distributed SLA Monitoring and Enforcement in Service-oriented Systems. International Journal On Advances in Systems and Measurements **2**(1) (2009) 33–43